ELSEVIER
Discrete Applied Mathematics 69 (1996) 169–182
DISCRETE APPLIED MATHEMATICS

All-pairs-shortest-length on strongly chordal graphs

V. Balachandhran, C. Pandu Rangan *

Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, India.

Received 12 October 1994

Abstract

The all-pairs-shortest-length (APSL) problem has been quite rigorously studied on various graphs. On general graphs, solutions are known for both the unweighted and weighted cases. Recently Seidel (1992) showed that the APSL problem on unweighted graphs can be solved in $O(M(n)\log n)$ time with $O(n^2)$ space. Here $M(n)$ denotes the complexity of multiplying two matrices of small integers, which is currently $O(n^{2.376})$.

The APSP problem has been solved on the class of permutation graphs (a subclass of co-comparability graphs) with the complexity $O(n+T)$ where T is the sum of the lengths of all the shortest paths (Mahesh, 1989). On the classes of interval and circular-arc graphs, an optimal solution to the APSL problem is available for the weighted case (Atallah et al; 1993).

In this paper, we discuss the APSL problem on the class of *strongly chordal graphs*. There is no algorithm known for the APSL problem on this class of graphs. We present an $O(n^2)$ algorithm to compute the lengths of the shortest-paths that also enables us to retrieve the paths in optimal time. The solution is for unweighted strongly chordal graphs.

Keywords: Shortest path; Strongly Chordal graph and Chordal graphs

1. Introduction

Let $G=(V,E)$ be an undirected, simple graph with no self-loops. Let $|V|=n$ and $|E|=m$. The following are some of the common shortest-path problems.

The *single-source shortest-length* (SSSL) problem is to find the length of the shortest-path from a given source vertex u to all other vertices. The *single-source shortest-path* (SSSP) problem is to find shortest paths from a given source vertex u to all other vertices. The *all-pairs-shortest-length* (APSL) problem is to find the length of the shortest path between all pairs of vertices. The *all-pairs-shortest-path* (APSP) problem is to find shortest paths between all pairs of vertices.

Table 1 presents the status of the APSL problem on a variety of graphs. $M(n)$ denotes the complexity of multiplying two $n \times n$ matrices of small integers, which is currently $O(n^{2.376})$.

* Corresponding author. E-mail: rangan@iitm.ernet.in.

SSDI 0166-218X(95)00088-7

Table 1

Restrictions on the graph	Time complexity	Reference
	$O(n^2m)$	[5]
	$O(n^3)$	[10]
	$O(mn + n^2 \log n)$	[11]
Without negative cycles	$O(n^3)$	[7]
Non-negative cycles	$O(mn + n^2 \log n)$	[9]
Unweighted	$O(M(n) \log n)$	[18]
Unweighted	$O(mn)$	[14]
DAG	$O(mn + n^2)$	[12]
Planar	$O(n^{1.5} \log n)$	[15]
Planar, non-negative weights	$O(n\sqrt{\log n})$	[8]
Unweighted Trapezoidal	$O(n^2)$	[2]
Interval	$O(n^2)$	[1]
Circular-arc	$O(n^2)$	[1]

The APSP problem has been solved on the class of permutation graphs (a subclass of cocomparability graphs) with the complexity $O(n + T)$ where T is the sum of the lengths of all the shortest paths [13].

In this paper, we discuss the APSL problem on the class of *strongly chordal graphs*. There is no algorithm known for the APSL problem on this class of graphs. We present an $O(n^2)$ algorithm to compute the lengths of the shortest paths that also enables us to retrieve the paths in optimal time.

The length of a path is denoted by the number of edges in the path. We use the notation $|P|$ to denote the number of vertices in the path P. $N_G(v)$ for a given vertex v in G denotes the open neighborhood of v in G. $N_G[v]$ denotes the closed neighborhood of v in G. When the context is clear, the subscript G can be dropped.

An edge is a *chord* of a cycle if it connects two vertices of the cycle but is not itself an edge within the cycle. A graph is *chordal* if and only if every cycle of length greater than three has a chord. Assume that an even cycle is numbered as $(v_1, v_2, \ldots, v_{2k})$ in the clockwise direction where $2k$ denotes the length of the cycle. Then, an edge (v_i, v_j) is an *odd chord* of an even cycle if $i-j$ is odd. A graph is *strongly chordal* if it is chordal and every even cycle of length six or more has an odd chord. Some more useful properties that characterize these graphs are given below.

A vertex v is called *simplicial* if the subgraph induced by $N[v]$ is a clique. Rose [17] showed that a graph G is chordal if and only if it is possible to order the vertices $(v_1, v_2, \ldots, v_n)$ in such a way that, for each $i \in \{1, 2, \ldots, n\}$, v_i is a simplicial vertex of G_i, where G_i is the subgraph induced by the vertex set $\{v_i, v_{i+1}, \ldots, v_n\}$. Such an ordering is called a *perfect elimination ordering* (also referred as the PE ordering).

Let $N_i[v]$ denote the closed neighborhood of v in G_i. The ordering of vertices $(v_1, v_2, \ldots, v_n)$ is called a *strong elimination ordering* (also referred as the SE ordering), if it is a perfect elimination ordering and for each $i < j < k$, if $v_j, v_k \in N_i[v_i]$, then $N_i[v_j] \subseteq N_i[v_k]$. In other words, for each vertex v_i, we not only require that it be simplicial in G_i, but in addition insist that the ordering of the vertices in $N_i[v_i]$

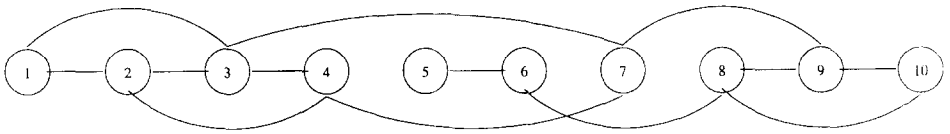

Fig. 1. A strongly chordal graph.

be consistent with the ordering of the corresponding cliques on increasing cardinality. Farber [6] defines a graph as being strongly chordal if it admits a strong elimination ordering and then shows that this is in fact equivalent to the previous definition in terms of odd chords.

The class of strongly chordal graphs is an interesting subclass of chordal graphs which properly includes trees, powers of trees and directed path graphs. Farber designed an $O(n^5)$ algorithm to recognize a strongly chordal graph which constructs the strong elimination ordering if one exists. Moreover, the ordering can be found in linear time for trees, powers of trees (when the underlying tree is known) and directed path graphs (given an appropriate path representation). To date, the best known algorithm to recognize a strongly chordal graph takes $O(m \log n)$ time. This algorithm additionally determines a strong elimination ordering [16].

We assume that the given graph is connected and that a strong elimination ordering is available for the given graph. In this paper, we use i to denote v_i, and henceforth $i < j$ iff v_i comes before v_j in the strong elimination ordering.

Example. Fig. 1 gives a strongly chordal graph, in which the vertices have been numbered according to a strong elimination ordering of the graph. For example, consider the vertex 2 in the graph. We have $3, 4 \in N_2[2]$ and hence $(3,4) \in E$. From the figure $N_2[3] = \{2, 3, 4, 7\}$ and $N_2[4] = \{2,3,4,7\}$. Since $N_2[3] \subseteq N_2[4]$, it satisfies the special property of the strong elimination ordering. Here 2 is a simplicial vertex as $N_2[2]$ induces a clique in the original graph.

2. Some properties of the shortest paths

Definition 2.1. Let $NEXT[i]$ denote the largest vertex that is adjacent to i. Let S_{ij} denote the set of all vertices reachable from i by a path of length $j-1$. That is

$$S_{ij} = \{x | \exists \text{ a path } P[i,x] \text{with } |P| = j\}$$

Let t_i be the smallest j such that S_{ij} contains the vertex n. We define

$$MAX[i,j] = \begin{cases} max(S_{ij}) & \text{if } j \leqslant t_i, \\ NULL & \text{otherwise.} \end{cases}$$

Intuitively, $MAX[i,j]$ denotes the largest vertex that is reachable from i by a path of length $j-1$ with the additional condition that there exists no shorter path from i to n.

Table 2
The array *NEXT* for the graph in Fig. 1

Vertex	1	2	3	4	5	6	7	8	9	10
NEXT	3	4	7	7	6	8	9	10	10	

Table 3
The array *MAX* for the graph in Fig. 1

MAX	1	2	3	4	5
1	1	3	7	9	10
2	2	4	7	9	10
3	3	7	9	10	
4	4	7	9	10	
5	5	6	8	10	
6	6	8	10		
7	7	9	10		
8	8	10			
9	9	10			
10	10				

Tables 2 and 3 respectively give the *NEXT* and *MAX* arrays for the graph of Fig. 1.

Lemma 2.1. *Let P be any shortest path from i to j, $i < j$. Let x, y and z be any three consecutive vertices in P in that order. Then neither $(y < z < x)$ nor $(y < x < z)$. That is, y will never be the smallest of the three vertices.*

Proof. Assume that x, y, z occur in that order in a shortest path P from i to j, with either $y < z < x$ or $y < x < z$. Recall that $N_y[y]$ denotes the closed neighborhood of y in the subgraph induced by $G_y = \{y, y+1, \ldots, n\}$. Since the SE ordering is also a PE ordering, $x, z \in N_y[y]$. We can see that $(x, z) \in E$ since $N_y[y]$ is clique. This implies that $P/\{y\}$ is a shorter path from i to j, a contradiction. Hence the lemma. □

Lemma 2.2. *For all $i < n$, there exists a j such that $j > i$ and $(i, j) \in E$.*

Proof. Assume that there exists no $j > i$ such that $(i, j) \in E$. Let $i' > i$. Since the graph is connected, there exists a path between i and i'. Let P be a shortest path from i to i'. Now we prove that every vertex v in P is greater than i.

If there are vertices smaller than i in P, consider the smallest among them, say y. Consider the two neighbors x and z of y in P.

Clearly x, y, z satisfy the condition that $(y < x < z)$ or $(y < z < x)$, a contradiction to Lemma 2.1. Hence no vertex in the shortest path P from i to i' is smaller than i. Thus i has a larger neighbor, which proves the lemma. □

Lemma 2.3. *For all i,j, if $MAX[i,j]$ and $MAX[i,j+1]$ are non-NULL, then $MAX[i,j] < MAX[i,j+1]$.*

Proof. From Lemma 2.2 we have a vertex k such that $k > MAX[i,j]$ and $(k, MAX[i,j]) \in E$. From the definition of MAX, $MAX[i,j+1] \geqslant k > MAX[i,j]$. Hence the lemma. □

Lemma 2.4. *For all i, when $MAX[i,j]$ and $MAX[i,j+1]$ are both non-NULL, we have $(MAX[i,j], MAX[i,j+1]) \in E$.*

Proof. We prove the lemma by induction on the length of the shortest paths from i. We use the notation z_j to denote the jth vertex in the path under consideration.

Induction base: *Let $2 \leqslant j < 4$ and x be a vertex satisfying $MAX[i,j-1] < x \leqslant MAX[i,j]$. Let there exist a shortest path of length $j-1$ from i to x,*

1. *There exists a shortest path from i to x, with the q^{th} vertex in the path, $z_q = MAX[i,q]$ for all $q \leqslant j-1$.*
2. *There cannot exist a shortest path from i to x such that $z_q < MAX[i,q-1]$ for any $q \leqslant j-1$.*
3. *$(MAX[i,j-1], MAX[i,j]) \in E$.*

The proof for the basis clause is given as follows.

Case 1: $j = 2$. Condition 1 is satisfied, since $z_1 = i$. Conditions 2 and 3 are trivially satisfied.

Case 2: $j = 3$. Here $MAX[i,2] < x < MAX[i,3]$. If $z_2 < i$, as $(i,z_2) \in E$ and $N_{z_2}[z_2]$ induces a clique in the given graph, we have $(x, MAX[i,1]) \in E$, a contradiction. Hence Condition 2 is satisfied. Condition 1 is satisfied for the following reasons. If $z_2 = MAX[i,2]$, Condition 1 is trivially satisfied. Hence let $i < z_2 < MAX[i,2]$. Since $(i, MAX[i,2]) \in E$, we have $(z_2, MAX[i,2]) \in E$. This implies that $(MAX[i,2], x) \in E$. The above facts are evident from the properties of the SE ordering. We see that, once Condition 1 is satisfied, Condition 3 is also satisfied, as x can be $MAX[i,j]$. Hence the basis clause stands validated.

Induction hypothesis: *Let $j \leqslant k$ and x be a vertex satisfying $[i,j-1] < x \leqslant \mathrm{MAX}[i,j]$. Let there exist a shortest path of length $j-1$ from i to x,*

1. *There exists a shortest path from i to x, with the qth vertex in the path, $z_q = MAX[i,q]$ for all $q \leqslant j-1$.*
2. *There cannot exist a shortest path from i to x such that $z_q < MAX[i,q-1]$ for any $q \leqslant j-1$.*
3. *$(MAX[i,j-1], MAX[i,j]) \in E$.*

Induction: *Now we prove the above three statements for the case $j = k+1$.*

From the induction hypothesis, it is clear that $(MAX[i,j-1], MAX[i,j]) \in E$, where $j \leqslant k$. Now consider some vertex $x > MAX[i,k]$, such that there exists a shortest path

of length k from i to x. Clearly $MAX[i,k] < x \leqslant MAX[i,k+1]$ from the definition of MAX.

If the penultimate vertex, i.e. z_k is such that $z_k < MAX[i,k-2]$, then $z_{k-1} < MAX[i, k-2]$ from lemma 2.1, a contradiction to the induction hypothesis, since for no q, z_q can be such that $z_q < MAX[i,q-1]$. Hence $z_k \geqslant MAX[i,k-2]$.

If $z_k = MAX[i,k-2], MAX[i,k-1] \geqslant x$, a contradiction. Hence $z_k > MAX[i,k-2]$.

Let $MAX[i,k-2] < z_k \leqslant MAX[i,k-1]$. If $z_{k-1} > MAX[i,k-2]$ (as $z_{k-1} \leqslant MAX[i, k-2]$ is ruled out), we see that $(z_{k-1}, MAX[i,k-2]) \in E$ from the induction hypothesis. Since $MAX[i,k-2] < z_{k-1} < z_k < MAX[i,k-1] (z_k = MAX[i,k-1]$ is out of question as we are considering a shortest path), and $(MAX[i,k-2], MAX[i,k-1]) \in E$, we have $(z_{k-1}, MAX[i,k-1]) \in E$. This is because $N_l[l]_{l=MAX[i,k-2]}$ induces a clique. We now observe that $(z_k, MAX[i,k-1]) \in E$. Since $(z_k, x) \in E$, we can see that $(MAX[i,k-1], x) \in E$. This implies that $MAX[i,k] \geqslant x$, a contradiction. Hence $MAX[i,k-1] < z_k \leqslant MAX[i,k]$. See Fig. 3 for this case. In the figure, $[i,j]$ represents $MAX[i,j]$, and an edge between vertices x and y is given by a curve with the end-points at x and y.

From the induction hypothesis, $(MAX[i,k-1], z_k) \in E$, and so $(z_k, MAX[i,k]) \in E$. Hence $(MAX[i,k], x) \in E$, as $N_{z_k}[z_k]$ induces a clique. Since the above argument holds for any x such that $MAX[i,k] < x \leqslant MAX[i,k+1]$, it follows that $(MAX[i,k], MAX[i, k+1]) \in E$.

From the above, we conclude that three conditions stated in the induction hypothesis hold for any arbitrary j.

The above proof can be very much simplified by considering the special property of the SE ordering, i.e. for $i < j < k$, if $v_j, v_k \in N_i[v_i]$, then $N_i[v_j] \subseteq N_i[v_k]$. But then such a proof does not apply to the class of chordal graphs. We have endeavored to show that most of the important properties observed on the class of strongly chordal graphs with respect to this problem, also hold on the class of chordal graphs. Such a general approach may contribute to subsequent attempts to solve the same problem on the class of chordal graphs, for which efficient solutions are not known till now.

The above lemma is illustrated by Table 3, where $(MAX[i,j], MAX[i,j+1]) \in E$ for all valid j.

Lemma 2.5. *Let P be any shortest path from i to $j, i < j$. Let j' be the first vertex in P greater than j. The path from i to j' in P, denoted by P', has its vertices in the increasing order of the SE ordering.*

Proof. We shall consider the vertices in P' in the order of its traversal from i to i'. Thus, we can talk of *next* and *previous* with respect to a vertex in P' unambiguously. Now, we have to show that $x > previous(x)$ for all x in P'. It is easily seen that $j' > j > x$ for any x in P'.

Let if possible, x' be the first vertex with $x' < previous(x')$. Let P'' denote the path from $previous(x')$ to j' in P'. Since $x' < j'$, P'' should contain a vertex y such that

$y < next(y)$. Let x'' be the first vertex in P'', such that $x'' < next(x'')$. Now, clearly, either $(x'' < previous(x'') < next(x''))$ or $(x'' < next(x'') < previous(x''))$, which contradicts Lemma 2.1, as P is a shortest path. Thus there does not exist any vertex x' in P', such that $x' < previous(x')$. Hence the lemma. □

Lemma 2.6. *Let $i < j$ and k be the integer satisfying $MAX[i,k] < j < MAX[i,k+1]$. Then there exists a shortest path from i to j containing $MAX[i,k]$.*

Proof. Let l be the first vertex greater than $MAX[i,k]$ in some shortest path P from i to j. Note that the vertex j may serve as a candidate for l. From Lemma 2.5, the path P' from i to l in P is a shortest path with the vertices in the increasing order of the SE ordering.

Assume that the length of P' is k. From the inductive proof for Lemma 2.4, we note that the rth vertex in P', z_r, is such that $z_r > MAX[i,r-1]$ and $(z_r, MAX[i,r-1]) \in E$, where $r \leqslant k+1$. We can see that $(MAX[i,k], z_k) \in E$, when $MAX[i,k-1] < z_k < MAX[i,k]$, since $MAX[i,k], z_k \in N_q[q]$, where $q = MAX[i,k-1]$. Otherwise $z_k = MAX[i,k]$. In the latter case, the lemma is true trivially. In the former case since $(z_{k+1}, z_k) \in E$, we have $(MAX[i,k], z_{k-1}) \in E$ and hence there is a shortest path from i to j containing $MAX[i,k]$. Now it is also clear that $z_{k+1} = l$ since otherwise, P' is not a shortest path from Lemma 2.1.

The other case to be handled is when the length of P' is greater than k. Clearly, there exists a q such that $z_q < MAX[i,q-1]$, where z_q denotes the qth vertex in the path P'. Let z_r be the first vertex in P', such that $z_r < MAX[i,r-1]$. Since P' monotonically increases according to the SE ordering, $z_{r-1} < z_r$. Clearly $MAX[i,r-2] < z_{r-1} < z_r < MAX[i,r-1]$. From the definition of z_r, there exists a path of length $r-2$, from i to z_{r-1}. From the inductive proof of Lemma 2.4, we see that $(MAX[i,r-2], z_{r-2}) \in E$. Since $(MAX[i,r-2], MAX[i,r-1]) \in E$, we have $(z_{r-1}, MAX[i,r-1]) \in E$. This implies that $(z_r, MAX[i,r-1]) \in E$. Now, as $z_r < z_{r+1}$, we have $(z_{r+1}, MAX[i,r-1]) \in E$. This in turn implies that z_{r+1} can be reached by a path of length $r-1$, a contradiction to the fact that P' is a shortest path from i to l.

Hence we see that for all $q, MAX[i,q-1] < z_q \leqslant MAX[i,q]$. This implies that the length of P' is k, in which case we have already proved that there exists a shortest path from i to j containing $MAX[i,k]$. □

Example. Consider the shortest path $\{1,3,7,9,8,6,5\}$ from vertex 1 to vertex 5 in the graph of Fig. 1. The first vertex that is greater than vertex 5 in the path is 7. The path $\{1,3,7\}$ is one where the vertices are in the increasing order of the SE ordering. This illustrates Lemma 2.5. Note that $MAX[1,2] < 5 < MAX[1,3]$. We see that the above shortest path contains the vertex 3, illustrating the above Lemma.

Note 2.1. We observe that the class of chordal graphs, a superset of this class, has similar properties as specified by Lemmas 2.2–2.6, since we have avoided using any attribute of strongly chordal graphs which does not hold for chordal graphs.

Lemma 2.7. *Let $i < j$, and j satisfy the condition that $MAX[i,k] < j < MAX[i,k+1]$. If $(MAX[i,k],j) \notin E$, there exists a shortest path from i to j containing $MAX[i,k+1]$.*

Proof. From Lemma 2.6, there exists a shortest path P from i to j containing $MAX[i,k]$. Since $(MAX[i,k],j) \notin E$, P contains some $v > MAX[i,k]$, such that v is next to $MAX[i,k]$ in P. Note that $v > MAX[i,k]$ from Lemma 2.1. It is clear that $v \leqslant MAX[i, k+1]$. If $v = MAX[i,k+1]$, the Lemma is proved. Otherwise, when v, $MAX[i,k+1] \in N_l[l]$ where $l = MAX[i,k]$. Hence from the property of the SE ordering, $N_l[v] \subseteq N_l[MAX[i,k+1]$. Hence $MAX[i,k+1]$ can be substituted for v in P to get P' which is also a shortest path from i to j. □

Note 2.2. We observe that Lemma 2.7 does not hold for the class of chordal graphs.

Example. In the illustrative example given for Lemma 2.6, note that $(MAX[i,2],5) \notin E$ and hence $MAX[i,3]$ is in the shortest path $\{1,3,7,9,8,6,5\}$ from vertex 1 to vertex 5.

3. Design of the algorithm

We begin by computing a shortest path between j and $MAX[i,k+1]$ where $MAX[i, k] < j < MAX[i,k+1]$, for all possible values of i,j and k. Then according to Lemmas 2.6 and 2.7, either the shortest path between i and j has $MAX[i,k]$ as the penultimate vertex or is the shortest path between i and $MAX[i,k]$ followed by the shortest path between $MAX[i,k+1]$ and j.

Let M_i denote the ith largest vertex such that $NEXT[v] = M_i$ for some vertex v. Clearly $M_1 = n$. Let L_i denote the smallest vertex such that $NEXT[L_i] = M_i$, with $L_i = NEXT[u]$ for some vertex u. Table 4 gives the arrays M and L for the graph of Fig. 1. Note that the vertices 1, 2 and 5 are not M_i for any i.

Let j be a vertex such that $MAX[i,k] < j < MAX[i,k+1] = M_q$ for some q.

Case 1: $((M_q,j) \in E)$. Any shortest path between j and M_q is of length one.

Case 2: $(NEXT[j] < M_q)$. A shortest path between j and M_q is any shortest path between $NEXT[j]$ and M_q preceded by the vertex j.

Case 3: $(NEXT[j] > M_q)$.

Table 4
The arrays M and L for the graph of Fig. 1

Index	1	2	3	4	5	6	7
M	10	9	8	7	6	4	3
L	8	7	6	3	5	2	1

If case 1 is not applicable, the shortest path is defined as in case 2. Here $NEXT[j]$ is some M_p where $p < q$.

The above analysis suggests a method of computing a shortest path between j and $MAX[i,k+1]$. To handle case 3, we compute the shortest path between j and M_q in the ascending order of q, such that $L_q < j < M_q$. To handle case 2, we compute a shortest path between j and M_q such that j takes values in the descending order of the SE ordering.

Algorithm shortest-length

Input: Some representation of a given strongly chordal graph with the vertices numbered in accordance with an SE ordering.

Output: $LSP[i,j]$, denoting the length of a shortest path from i to j and $S[i,j]$ denoting the vertex next to i in some shortest path from i to j.

```
/* If A and B are paths, A.B denotes the concatenation of the paths. */
/* SP[i,j] denotes a shortest path between i and j. */
/* The arrays NEXT, M and L are assumed to be available. */
begin
    q = 1;
    while (M_q ≠ NULL)
    begin
        v = M_q − 1;
        while (v > L_q)
        begin
            if (v, M_q) ∈ E then
            begin
                LSP[v, M_q] = 1;
                S[v, M_q] = M_q;
                /* SP[v, M_q] = {v, M_q} */
            end
            else
            begin
            /* Lemmas 2.6 and 2.7 applied here */
                LSP[v, M_q] = LSP[NEXT[v], M_q] + 1;
                S[v, M_q] = NEXT[v];
                /* SP[v, M_q] = (v).SP[NEXT[v], M_q] */
            end
            v = v − 1;
        end
        q = q + 1;
    end
```

```
    for i = 1 to n − 1
    begin
        k = 2; j = i + 1;
        while (MAX[i,k] ≠ NULL)
        begin
            while (j ⩽ MAX[i,k])
            begin
                if j = MAX[i,k] or (j, MAX[i,k − 1]) ∈ E then
                begin
                    LSP[i,j] = k − 1;
                    S[i,j] = NEXT[i];
                    /* SP[i,j] = SP[i, MAX[i,k − 1]].(j) */
                end
                else
                begin
                    LSP[i,j] = LSP[i, MAX[i,k]] + LSP[MAX[i,k], j];
                    S[i,j] = NEXT[i];
                    /* SP[i,j] = SP[i, MAX[i,k]].(SP[MAX[i,k], j]/MAX[i,k]) */
                end
                j = j + 1;
            end
            k = k + 1;
        end
    end
end
```

An example showing the execution details of the algorithm

Consider the graph of Fig. 1.

1. $M_1 = 10$; $L_1 = 9$;
   There are no vertices between M_1 and L_1.
2. $M_2 = 9$; $L_2 = 7$;
   There is the vertex 8 between vertices 7 and 9.
   $LSP[8,9] = 1$; $S[8,9] = 9$;
3. $M_3 = 8$; $L_3 = 6$;
   The only intermediate vertex is the vertex 7.
   $(7,8) \notin E$. Hence $LSP[7,8] = LSP[7,NEXT[7]] + LSP[NEXT[7],8]$;
   $= LSP[7,9] + LSP[9,8] = LSP[7,9] + LSP[8,9] = 1 + 1 = 2$;
   $S[7,8] = 9$;
4. $M_4 = 7$; $L_4 = 3$
   The intermediate vertices are {4, 5, 6}

Table 5
The array LSP for the graph of Fig. 1

LSP	1	2	3	4	5	6	7	8	9	10
1		1	1	2	6	7	2	4	3	4
2			1	1	6	5	2	4	3	4
3				1	5	4	1	3	2	3
4					5	4	1	3	2	3
5						1	4	2	3	3
6							3	1	2	2
7								2	1	2
8									1	1
9										1
10										

Table 6
The array S for the graph of Fig. 1

S	1	2	3	4	5	6	7	8	9	10
1		2	3	3	3	3	3	3	3	3
2			3	4	4	4	4	4	4	4
3				4	7	7	7	7	7	7
4					7	7	7	7	7	7
5						6	6	6	6	6
6							8	8	8	8
7								9	9	9
8									9	10
9										10
10										

$(6,7) \notin E$ and similarly as in step 3 $LSP[6,7] = LSP[6,8] + LSP[7,8]$

Hence $LSP[6,7] = 1 + 2 = 3$; $S[6,7] = 8$;
$(5,7) \notin E$. So $LSP[5,7] = LSP[5,6] + LSP[6,7] = 1 + 3 = 4$; $S[5,7] = 6$;
$LSP[4,7] = 1$ and $S[4,7] = 7$;

5. $M_5 = 6$; $L_5 = 5$;
   We have no intermediate vertices.
6. $M_6 = 4$; $L_6 = 2$;
   Vertex 3 is the only intermediate vertex.
   $LSP[3,4] = 1$; $S[3,4] = 4$;
7. $M_7 = 3$ and $L_7 = 1$;
   The only intermediate vertex is vertex 2.
   $LSP[2,3] = 1$; $S[2,3] = 3$;

Using the above computation, the rest of the shortest paths can easily be found out and hence they are not worked out here. Table 5 and Table 6 give respectively, the arrays LSP and S for the graph of Fig. 1.

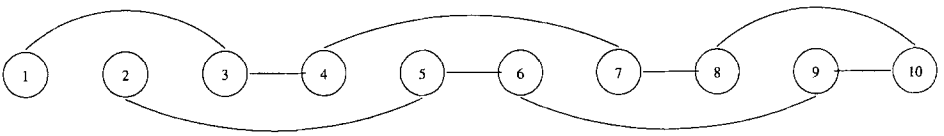

Fig. 2.

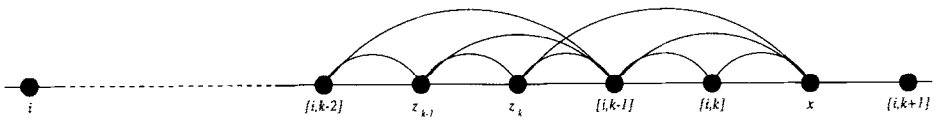

Fig. 3.

It is clear that Fig. 1 gives an example of a typical strongly chordal graph in which all the cases occurring in the algorithm are put to use. Fig. 2 gives an extreme example in which there exists only one shortest path between consecutive vertices 1 and 2 and the path is $\{1,3,4,7,8,10,9,6,5,2\}$. In this example too, the numbering of the vertices is according to a strong elimination ordering of the graph. Hand-simulating the above algorithm on the graph of Fig. 3, we can see more clearly as to why the computation proceeds in the ascending order of the M_i's and then in the descending order of the vertices between any L_j and M_j.

4. Correctness and complexity analysis of algorithm shortest length

The arrays *NEXT*, *M* and *L* can be easily computed in $O(n^2)$ time. It is clear that by the time the algorithm sees a shortest path as being made of two shortest paths, both the shortest paths would have been computed, since the algorithm follows the observation in the previous section. Because of this, the time taken for the second **while** loop ($v > L_q$), to execute once is constant. It is evident that less than n^2 pairs are considered by the first two **while** loops. Each execution of the innermost **while** loop inside the **for** loop takes only constant time again. Since for every i, only $n-i$ values are taken by j, each execution of the **for** loop takes a maximum of $n-i$ steps. Hence the time complexity of the algorithm is $O(n^2)$. The space complexity is dictated by the arrays *MAX*, *S* and *LSP* which again is $O(n^2)$. The computed shortest paths can be retrieved using the array *S*. This way any shortest path can be retrieved instead of computing all the $O(n^2)$ shortest paths at the same time. Consequent to the above discussion, we state the following theorem.

Theorem 4.1. *Given a strongly chordal graph and a strongly elimination ordering of the graph, the all-pairs-shortest-length problem can be solved in* $O(n^2)$ *time and space.*

5. Conclusion

This paper developed a solution to the APSL problem on the class of strongly chordal graphs. The task that we set out to achieve is a solution to the shortest-length query problem. To this end, we tried to apply the same paradigm that runs through the solutions to the problem on the classes of interval, circular-arc and permutation graphs [4, 3]. In other words, our stress was on constructing some kind of a path-tree where the shortest paths are embedded, such that the problem is reduced to that of level-ancestor queries on the path-tree. It is still possible to embed the shortest paths in a path-tree using the $NEXT$ function, but we found the task of finding the length of the shortest paths with constant number of level-ancestor queries, a very difficult one, as the following example will illustrate.

Consider the strongly chordal graph of Fig. 2. Let the SE ordering used by our algorithm be in the increasing order of the vertices. Clearly this is an SE ordering of the given graph. Assume that we want to find the length of any shortest path between vertices 1 and 2, which are consecutive vertices according to the SE ordering. Let $SP[i,j]$ denote any shortest path from i to j. From Lemma 2.7, there exists an $SP[1,2]$ with vertex 3 coming after vertex 1, as $NEXT[1]=3$. Hence $SP[1,2]$ can be expressed as $SP[1,3]$ followed by $SP[3,2]$. Since $NEXT[2]=5$, $SP[2,3]$ can be expressed as $SP[2,5]$ followed by $SP[3,5]$ and so on. If a path-tree is constructed with the $NEXT$ function, as in the case of the other solutions, then a single path-query may result in more path-queries thereby preventing a constant time answer to the query. In fact it is quite obvious that a single query could result in $O(n)$ queries, since we can construct SE orderings in which the only edges are between vertices separated by a single vertex in the SE ordering. It is to be noted that all these observations are important only if there is a way of marking off the first vertex that is greater than j, in the path from i to the root of the path-tree, when we want to find a shortest path between i and j with $i<j$. We can easily see that another SE ordering for the graph in Fig. 2 is $\{1,3,4,7,8,10,9,6,5,2\}$, which does not have any "problematic" paths, as in the previous SE ordering. So, preprocessing the given SE ordering to eliminate the "problematic" paths and trying for a solution to the shortest path query problem might be a promising venture.

We showed that Lemmas 2.1 to 2.6 apply to the class of chordal graphs as well. However, it is Lemma 2.7 that has resulted in the algorithm on strongly chordal graphs. In the light of the above, future research on the APSL problem on chordal graphs might need to adopt a different perspective from the one presented here.

References

[1] M.J. Atallah, D.Z. Chen and D.T. Lee, An optimal algorithm for shortest paths on weighted interval and circular-arc graphs with applications, in: Proceedings 1st Annual European Symposium on Algorithms Bonn, Germany (1993), Algorithmica 14(5) (1995) 429–441.

[2] V. Balachandran and C. Pandu Rangan, An approximate algorithm for shortest-length queries on trapezoidal graphs, Technical Report TR-TCS-94-09, Dept. of CS & E, Indian Institute of Technology, Madras-36 (1994).
[3] V. Balachandran and C. Pandu Rangan, Shortest-length queries on interval and circular-arc graphs, Technical Report TR-TCS-94-10, Dept. of CS & E, Indian Institute of Technology, Madras-36 (1994).
[4] V. Balachandran and C. Pandu Rangan, Shortest-length queries on permutation graphs, Technical Report TR-TCS-94-08, Dept. of CS & E, Indian Institute of Technology, Madras-36 (1994).
[5] T.H. Cormen, C.E. Leiserson and R.L. Rivest, Intoduction to Algorithms (The MIT press and McGraw-Hill, Cambridge MA, 1990).
[6] M. Farber, Characterization of strongly chordal graphs, Discrete Math. 43 (1983) 173–189.
[7] R.W. Floyd, Shortest path, Comm. ACM 5 (1962) 345.
[8] G.N. Frederickson, Fast algorithms for shortest paths in planar graphs, with applications, SIAM J. Comput. 16 (1989) 1004–1022.
[9] M.L. Fredman and R.E. Tarjan, Fibonocci heaps and their uses in improved network optimization algorithms, J. ACM 34 (1987) 596–615.
[10] G. Gallo and S. Pallotino, A new algorithm to find the shortest paths between all pairs of nodes, Discrete Appl. Math. 4 (1982) 23–35.
[11] D.B. Johnson, Efficient algorithms for shortest paths in sparse networks, J. ACM, 24 (1977) 1–13.
[12] E.L. Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston, 1976).
[13] R. Mahesh, Domination and path problems in permutation graphs, Thesis Report (1989).
[14] Udi Manber, Introduction to Algorithms – A Creative Approach (Addison-Wesley, 1989).
[15] K. Mehlhorn and B.H. Schmidt. A single source shortest path algorithm for graphs with separators, in: Proceedings FCT 83, Lecture Notes in Computer Science, Vol. 158 (Springer, Berlin, 1983) 302–309.
[16] R. Paige and R.E. Tarjan, Three partition refinement algorithms, SIAM J. Comput. 16 (1987) 973–989.
[17] D.J. Rose, Triangulated graphs and the elimination process, J. Math. Anal. 32 (1970) 597–607.
[18] R. Seidel, On all-pairs-shortest-path problem, in: Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, Canada (1992) 745–749.